SLAM
Introduction & Data Structures

EDAA - G06

João Costa
Henrique Ribeiro – João Martins
Tiago Duarte

Project description

Simultaneous Location And Mapping

- **Goal** – map an environment navigated by and autonomous vehicle, while simultaneously locating it in the map;
- **Challenges:**
  - No access to pre-existing maps or external devices;
  - Focus on sub-aquatic SLAM ⇒ difficult access and extra data noise;
- **Datasets** – the group will have access to sonar data measured by CRAS.

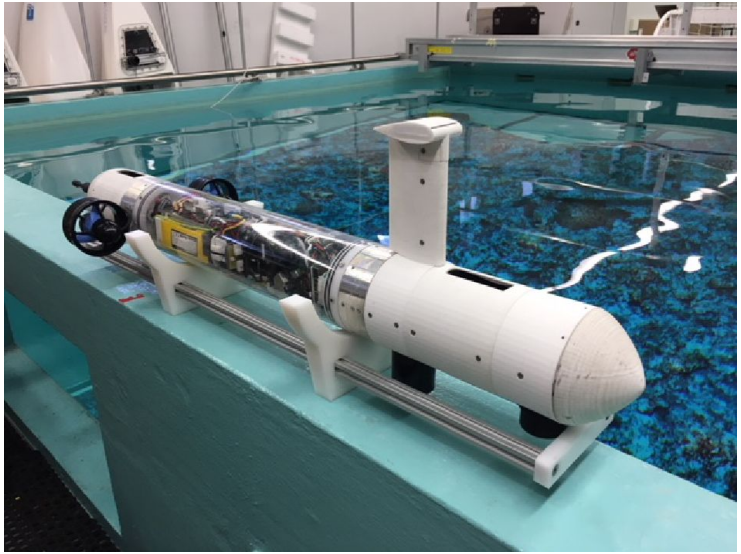

Fig 1. UAV used to collect the datasets.

Sonar data

- A sonar mounted on a vehicle collects environment data;
- Sonar data contains noise and other undesirable effects (e.g. multipath);
- **Fig 2.** Illustrates the raw data of the sonar and the problems present in this data:
    - Reflections from the body of the vehicle (self reflections);
    - Multipath effects when signals go through the tank walls;
    - Noise affecting detection of features (e.g. tank walls and floater).
- It is important to clean this data and find the distance to the first feature for each measurement.

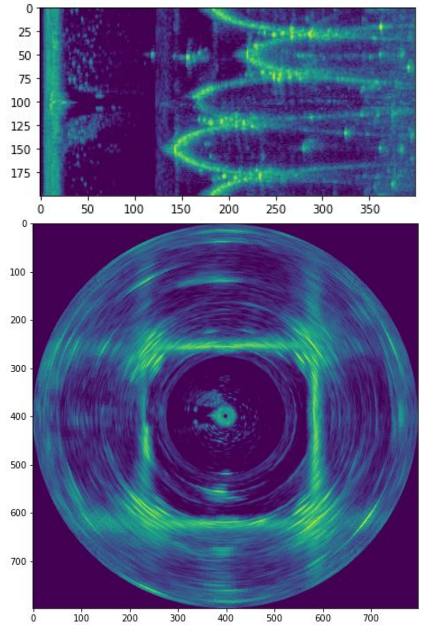

Fig 2. Dataset representation in polar and Cartesian coordinates.

Problems to be tackled

- Filter noise/undesirable effects in data:
    - Multiple reflections;
    - Echos;
    - Self reflections;
    - Multipath errors.
- Represent map in a space efficient format;
- Implement static/dynamic probabilistic mapping algorithms based on sonar data;
- Incorporate new measurements into the map according to some time constraints ($\approx$ 40ms).

Note: the group will focus on 2D mapping with a static vehicle in the first part of the project.

Data structure

Mapping representation alternatives

- **Occupancy grid –** the most commonly used representation;
- **Feature based maps –** mapping an area through points of interest in the data;
- **Octomaps –** efficient division of the map into cells.

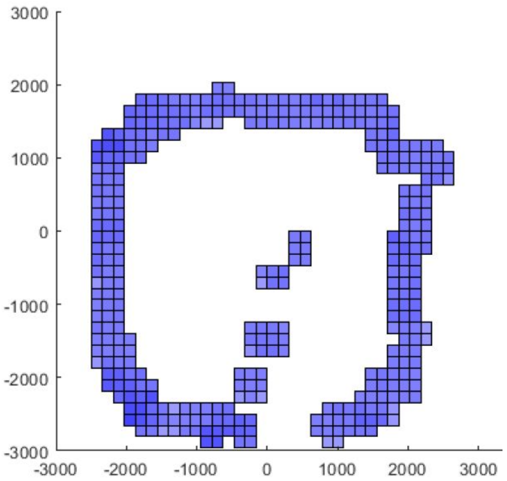

Fig 3. Mapping Representation

Occupancy Grid

- **Definition**:
  - Map the environment into a binary matrix (2 or 3D);
  - Usually, zeros represent unknown/occupied locations, and ones represent free cells.
- **Advantages**:
  - Easy to implement;
  - Represents the state directly.
- **Disadvantages**:
  - Fixed map size (doesn't scale);
  - Spatially inefficient;
  - Detail representation is limited.

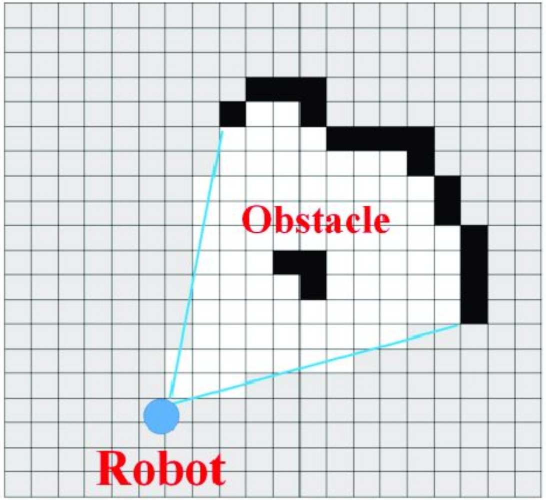

Fig 4. Occupancy Grid

Feature Based Map

- **Definition**:
  - Mapping an area through points of interest;
- **Advantages**:
  - Unique features help locate the vehicle;
- **Disadvantages**:
  - Features can be hard to identify;
  - Feature strength can be hard to identify;
  - Difficult to store the information efficiently;

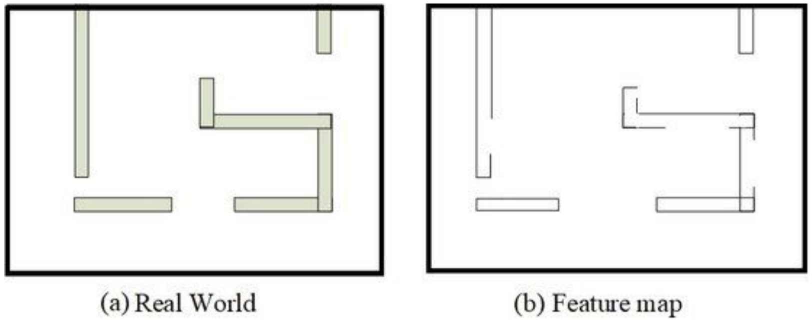

Fig 5. Feature Map

Octomap

- **Definition**:
  - The map is divided into cells that can be subdivided into 8 smaller nodes;
  - Based on **octrees**;
  - Division only happens when there's a probability of the voxel containing an object.
- **Advantages**:
  - Space efficient;
  - Recursive and always adapting.
- **Disadvantages**:
  - Implementations are harder.

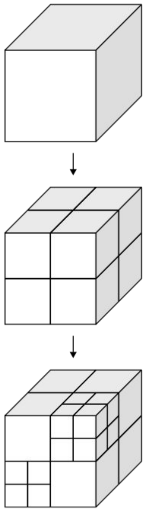

Fig 6. Octomap

Note: Although we're focusing on 2D for now, it was decided that it was better to implement Octomaps instead of Quadmaps, because we'll use 3D in the future.

Chosen data structure: Octree

- **Octrees** are a tree data structure where every internal (non-leaf) node has exactly 8 children;
- Useful to encode information during tree transversal in some applications: information compression;
- **Common supported operations**:

Node Traversal	$\mathcal{O}(n)$
Pruning	$\mathcal{O}(n)$
Search	$\mathcal{O}(\log n) = \mathcal{O}(d_{max})$
Insertion	$\mathcal{O}(\log n)$

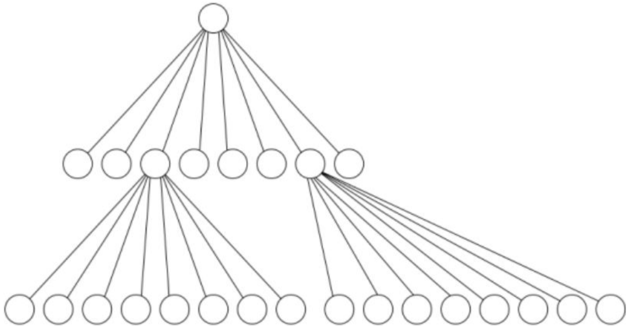

Fig 7. Octree

Octree - Example applications: 3D computer graphics

- By using **octrees**, it is possible to represent complex 3D objects in a compressed manner:
  - Detail can be controlled by choosing a max depth to the tree (limiting divisions).
- Can represent arbitrary objects;
- Image generation is efficient;
- Enables *view frustum culling* (Mee Young Sung et al.);
- Modern game engines like Unity have multiple implementations of octrees available.

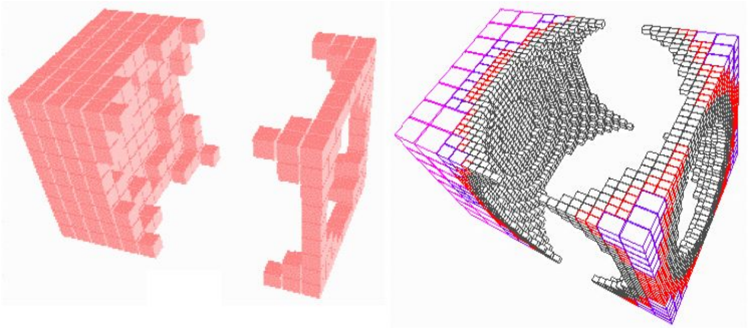

Fig 8. 3D object octree

Octree - Example applications: AllRGB

- AllRGB is a project where people create images with one pixel for every RGB color (without repetitions);
- A complete **octree** of 9 levels can be used to find approximations to colors that were already used:
    - Each node stores how many free children it has. Leafs are either 1 (used) or 0 (not used);
    - Nodes are updated during traversal.
- The color approximation is calculated during the traversal of the tree in 8 hops:
    - Algorithm explained at https://github.com/JoaoCostaIFG/allrgb

Fig 9. AllRBG example.

Octomaps/Octrees in our project

- **Octrees** will be used to implement **octomaps**:
  - The max depth of the underlying octree determines the max distance represented by the map. E.g., a max depth of 16 with a resolution of 1 cm allows for 655 m³.
- The **octomap** will be probabilistic:
  - 3 types of cells: free, occupied, and unknown;
  - Each cell has a probability of being empty associated (**log-odds**): 0.5 means unknown.
- Clamping update policy:
  - **Stable node –** when the **log-odds** value of a cell reaches a lower/upper bound;
  - In a static environment, it is likely that all nodes will become **stable**;
  - If all the children of a node are **stable** and agree on the occupancy state (free/occupied), then the children can be pruned.

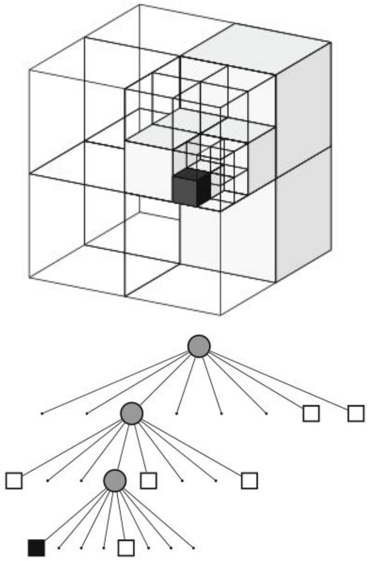

Fig 10. Example of octree storing occupied (black) and free (white) cells (Hornung et al.)

- The information in the **octomap** will be updated by ray-casting: discussed in the next presentation (algorithms);
- The map can be evaluated at a lower resolution for faster (but less detailed) drawing:
  - Each intermediate node’s occupancy is based on its eight children’s occupancy levels;
  - The maximum of those occupancy levels is used (instead of the average);
  - This leads to a pessimistic view of the free space, which is useful to avoid collision with the terrain by the vehicle.
- Space optimization:
  - Use a pointer to an array instead of 8 pointers.

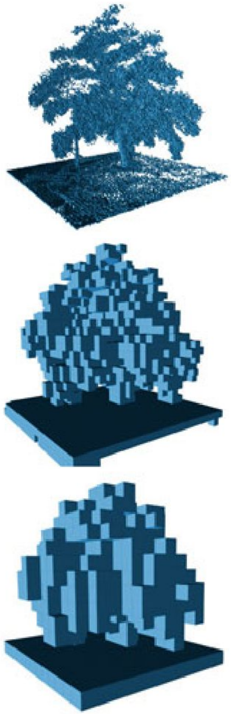

Fig 11. Example of octree depth limiting (Hornung et al.)

Questions?